SOAP - ENCODING

SOAP includes a built-in set of rules for encoding data types. It enables the SOAP message to indicate specific data types, such as integers, floats, doubles, or arrays.

- SOAP data types are divided into two broad categories − scalar types and compound types.
- Scalar types contain exactly one value such as a last name, price, or product description.
- Compound types contain multiple values such as a purchase order or a list of stock quotes.
- Compound types are further subdivided into arrays and structs.
- The encoding style for a SOAP message is set via the *SOAP-ENV:encodingStyle* attribute.
- To use SOAP 1.1 encoding, use the value http://schemas.xmlsoap.org/soap/encoding/
- To use SOAP 1.2 encoding, use the value http://www.w3.org/2001/12/soap-encoding
- Latest SOAP specification adopts all the built-in types defined by XML Schema. Still, SOAP maintains its own convention for defining constructs not standardized by XML Schema, such as arrays and references.

Scalar Types

For scalar types, SOAP adopts all the built-in simple types specified by the XML Schema specification. This includes strings, floats, doubles, and integers.

The following table lists the main simple types, excerpted from the XML Schema Part 0 − Primer http://www.w3.org/TR/2000/WD-xmlschema-0-20000407/

Simple Types Built-In to XML Schema	
Simple Type	**Example*s***
string	Confirm this is electric.
boolean	true, false, 1, 0.
float	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN.
double	-INF, -1E4, -0, 0, 12.78E-2, 12, INF, NaN.
decimal	-1.23, 0, 123.4, 1000.00.
binary	100010
integer	-126789, -1, 0, 1, 126789.
nonPositiveInteger	-126789, -1, 0.
negativeInteger	-126789, -1.
long	-1, 12678967543233
int	-1, 126789675
short	-1, 12678
byte	-1, 126
nonNegativeInteger	0, 1, 126789

unsignedLong	0, 12678967543233
unsignedInt	0, 1267896754
unsignedShort	0, 12678
unsignedByte	0, 126
positiveInteger	1, 126789.
date	1999-05-31, ---05.
time	13:20:00.000, 13:20:00.000-05:00

For example, here is a SOAP response with a double data type −

```
<?xml version='1.0' encoding='UTF-8'?>

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getPriceResponse xmlns:ns1="urn:examples:priceservice"  SOAP-
ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

         <return xsi:type="xsd:double">54.99</return>

      </ns1:getPriceResponse>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Compound Types

SOAP arrays have a very specific set of rules, which require that you specify both the element type and array size. SOAP also supports multidimensional arrays, but not all SOAP implementations support multidimensional functionality.

To create an array, you must specify it as an *xsi:type* of array. The array must also include an *arrayType* attribute. This attribute is required to specify the data type for the contained elements and the dimension*s* of the array.

For example, the following attribute specifies an array of 10 double values −

```
arrayType = "xsd:double[10]"
```

In contrast, the following attribute specifies a two-dimensional array of strings −

```
arrayType = "xsd:string[5,5]"
```

Here is a sample SOAP response with an array of double values −

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getPriceListResponse xmlns:ns1="urn:examples:pricelistservice"  SOAP-
ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

         <return xmlns:ns2="http://www.w3.org/2001/09/soap-encoding"
xsi:type="ns2:Array" ns2:arrayType="xsd:double[2]">
            <item xsi:type="xsd:double">54.99</item>
```

```
            <item xsi:type="xsd:double">19.99</item>
         </return>

      </ns1:getPriceListResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Structs contain multiple values, but each element is specified with a unique accessor element. For example, consider an item within a product catalog. In this case, the struct might contain a product SKU, product name, description, and price. Here is how such a struct would be represented in a SOAP message −

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://www.w3.org/2001/12/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <SOAP-ENV:Body>
      <ns1:getProductResponse xmlns:ns1="urn:examples:productservice" SOAP-
ENV:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

         <return xmlns:ns2="urn:examples" xsi:type="ns2:product">
            <name xsi:type="xsd:string">Red Hat Linux</name>
            <price xsi:type="xsd:double">54.99</price>

            <description xsi:type="xsd:string">
               Red Hat Linux Operating System
            </description>
            <SKU xsi:type="xsd:string">A358185</SKU>
         </return>

      </ns1:getProductResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

NOTE − Please take care of proper indentation while you write your SOAP code. Each element in a struct is specified with a unique accessor name. For example, the message above includes four accessor elements − name, price, description, and SKU. Each element can have its own data type. For example, name is specified as a string, whereas price is specified as double.